Benefits of Embedding *Smart*Charts™ within Developer/2000 Applications

Strategic White Paper

Oracle Developer/2000

ORACLE®

Table of Contents

Introduction

Oracle Forms developers can *easily* deliver powerful point-and-click applications by embedding graphical objects developed in Oracle Graphics 2.5. Oracle Graphics extends the capabilities of Oracle Forms 4.5 to offer interactive graphical controls that greatly enhance the usability and visual impact of Forms applications. This paper discusses the many benefits of integrating Oracle Graphics into Forms applications. These benefits include interactive graphical drill-down, fully customizable runtime behavior (via automation control), and automatic exception reporting.

Powerful Forms Applications

Using Oracle Graphics 2.5, developers can deliver Forms applications that present end users with a productive and inviting graphical interface. A clear strength of Oracle Forms is its ability to automatically establish and manage transactional activity between client and server. As a result, users have quick access to large quantities of data.

However, the productivity bottleneck is often the user's ability to extract useful information from their raw data. There may be significant portions of data that the user must identify immediately. Users may need to identify trends in the data or navigate quickly to more detailed sets of data. To alleviate the complexity of these tasks, developers search for ways to make their applications more graphical and interactive.

With the advent of Developer/2000, developers have the means to create these types of applications. By exploiting new features in Oracle Graphics 2.5, Forms developers can create dynamic graphical applications with easy-to-use point-and-click interfaces. Oracle Graphics adds tremendous value to Forms applications by offering a more direct and intuitive means of visualizing and interacting with data.

Oracle Graphics SmartCharts ä

Embedded Graphics objects are often referred to as SmartCharts because of their unique interactive, data-driven capabilities. No other products provide the same level of integration, flexibility, and responsiveness that SmartCharts deliver when embedded within Forms. With the new Forms OLE2 functionality, Forms developers have several alternatives for delivering graphical applications. However, limitations exist in the current OLE2 application interface that prevent developers from creating genuinely integrated applications. These limitations are discussed later to underscore specific benefits provided by Forms-Graphics integration. In addition, there are intrinsic limitations within these third party graphics servers that prevent Forms developers from creating truly data-driven graphical applications. Thus, SmartCharts offer Forms developers more functionality than embedded graphical objects from other products.

Graphical Drill-down

By embedding Oracle Graphics SmartCharts, developers can create applications with graphical drill-down. A graphical drill-down relationship is an association between a Forms block (or item) and an embedded Oracle Graphics SmartChart. The concept is very similar to traditional Forms master-detail relationships. Users browse data very quickly by drilling down from summary information to detail information. However, graphical drill-down offers several advantages over standard master-detail. First of all, users can generally observe trends or exceptions more quickly from a graphical representation of data than a textual representation. These quick observations often serve as the impetus for drilling

down and seeing more details. Another advantage is that SmartCharts deliver a *point-and-click* drill-down interface that can dramatically improve the usability of a Forms applications. Users can navigate *directly* to detail information by clicking on summary information. The examples described below serve to illustrate these points.

Chart to Block

One very common type of graphical drill-down involves a master-detail relationship between an embedded Oracle Graphics SmartChart and a Forms *detail* block. A graphical presentation of summary information often enables users to quickly identify areas that warrant further investigation. Because embedded SmartCharts are interactive, users drill down by clicking directly on the appropriate area of the chart. Details related to the selection are listed in nearby text items.

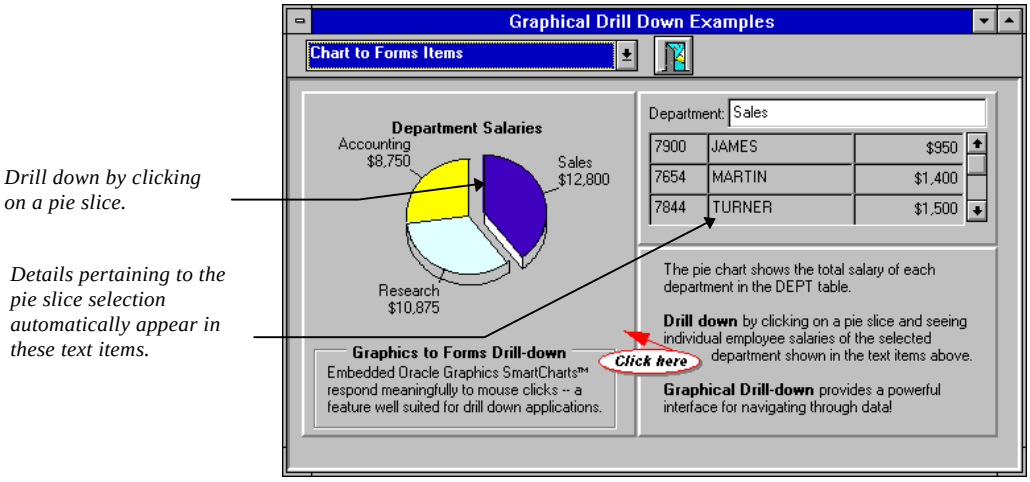

Figure 1.

In Figure 1, the pie chart shows total salaries of each department within a company. The user can compare the salary expenses of each department. If they would like to see more details about a specific department's salary expenses, they can click on the department's pie slice. This automatically updates the information in the Forms detail block. In general, developers can construct this sort of graphical drill-down out of any master-detail relationship. For example, a developer might wish to create a bar chart of warehouse inventories that allows users to click and drill down to see a detailed listing of the parts stored in each warehouse.

This example also illustrates the tight level of integration that exists between Forms and Graphics. The drill-down is coordinated between an embedded Graphics object and a Forms block. In this example, Forms sends chart data to Graphics. When a user clicks on the pie chart, Graphics sends Forms information about the selected pie slice. This information is then used by Forms to populate the text items with data pertaining only to the selected pie slice.

The ability to coordinate a drill-down between Forms and an embedded SmartChart is made possible through Graphics' support for bi-directional data passing. Forms can pass information to Graphics and receive information back from Graphics. **This bi-directional data exchange *cannot* occur between Forms and embedded OLE2 in-place activation servers**. OLE2 does not provide a mechanism for the embedded server application to communicate back to the container. Thus, it is not possible to coordinate a drill-down between an in-place activated OLE2 object and Forms. *Only* by using Graphics

SmartCharts can developers create applications in which the user has the ability to interact with an embedded graphical object and affect other objects on the form.

Map to Block

Figure 2 shows another example of an embedded Graphics object. The map of the United States leverages the user's familiarity with geographical locations, providing an intuitive interface for clicking and drilling down to see city population details.

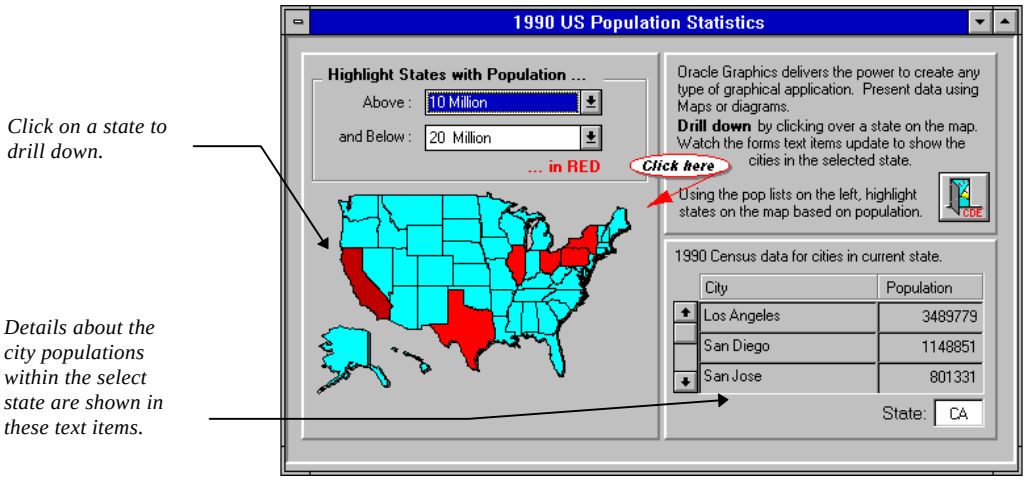

Figure 2.

In addition to its powerful charting capabilities, Graphics provides excellent support for creating *custom* graphical displays. As seen in Figure 2, Forms developers can embed interactive graphical displays that are maps or diagrams. Developers can import clip art or construct diagrams using Graphics' drawing tools. Graphics provides full programmatic support for creating objects dynamically and changing object properties (such as color) to reflect underlying data queried from the server. Graphics also delivers complete flexibility to create point-and-click drill-down objects out of custom drawings; since developers can make *any* drawing object (such as a polygon representing a geographical region) respond meaningfully to mouse events.

Block to Chart

Forms developers can also use embedded Graphics SmartCharts as the "detail" component in a master-detail relationship. By selecting a row from a "master" block, the user can see detail data populating an embedded SmartChart. This type of graphical drill-down provides a quick means of choosing the data used to populate an embedded chart.

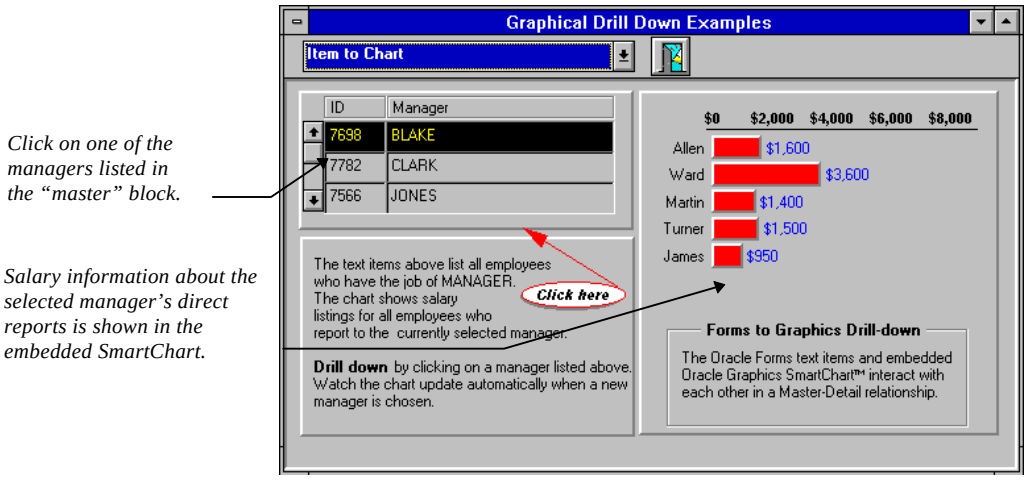

Figure 3.

Figure 3 shows a Forms application in which the user can click on a manager (listed in the "master" block) and see detail information displayed in a bar chart of employee salaries. By navigating to each item in the master block, the user can quickly browse information about managers and the salaries of their direct reports.

Chart to Chart

It is also possible to embed a single Oracle Graphics object containing two charts functioning in a master-detail relationship. In Figure 4, the pie chart displays head count information for each department within a company and the bar chart displays details about the salaries of each employee within a specific department. Users drill down by clicking on a pie slice; the bar chart updates automatically. In Oracle Graphics 2.5, developers can establish drill-down relationships between charts ***without writing a single line of code***.

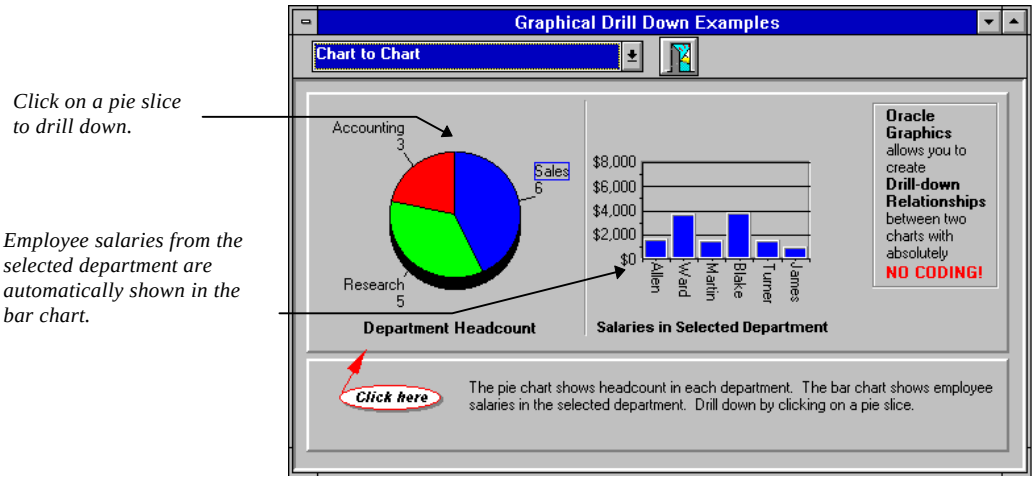

Figure 4.

Full Automation Control

Full automation control is Forms' ability to completely dictate the behavior of an embedded SmartChart. As an embedded graphics server, Graphics provides tremendous flexibility for customized interaction with Forms. From any Forms trigger, developers can call any one of the 400+ built-in procedures provided in Graphics. These built-ins allow developers to

dynamically change object properties, including color, pattern, position, and rotation. There are also built-ins for changing chart properties such as axis settings, reference lines, labels, and data field mapping. Furthermore, Forms developers can write their own *custom* Graphics procedures and control embedded SmartCharts by calling these custom procedures within their Forms applications.

While it is possible to embed other OLE2 servers within their forms, developers should note that there are often limitations on the extent to which they can customize the behavior of these embedded objects. To control the behavior of an embedded OLE2 object, developers rely on automation methods externalized by the OLE2 server. No OLE2 servers come close to providing application developers with the number of automation methods that are available to them when embedding Graphics. The limited scope of the automation methods is due, in part, to some restrictions inherent within the OLE2 application interface. In addition, developers often find it very difficult to register their own custom scripts, or procedures, as automation methods. Compared to Graphics, embedded OLE2 objects offer Forms developers very little flexibility for customized behavior of embedded objects.

The following examples illustrate a few ways that Forms developers can exercise full automation control over embedded Graphics SmartCharts.

Change Chart Type

Figure 5 shows a Forms application in which a user can dynamically change the chart type of an embedded SmartChart by using Forms iconic buttons or a Forms list item. Thus, the user has the flexibility to choose the best graphical presentation for their runtime data.

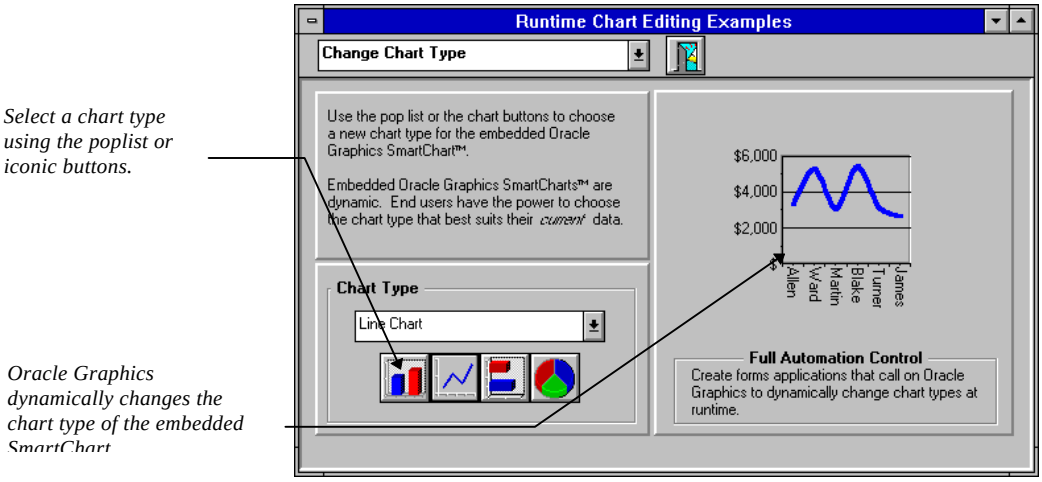

Figure 5.

This example is just one way a developer might choose to deliver chart editing capabilities within their Forms application. By using different Forms items and different Graphics chart editing built-ins, developers have the flexibility to construct any desired chart editing interface. This allows them to deliver much more intuitive Forms modules. Within a given application, users usually need to specify only a few properties of their chart. All other chart properties can (and should) be defaulted for the end user based on application context.

In these situations, providing full-blown chart editing capabilities may unnecessarily clutter the interface and overwhelm end users. Usability tests have shown that even with the best designed chart editing interfaces, many users find the task of editing a chart from scratch to be very intimidating. This may pose a usability problem for Forms developers. Nearly all

embedded OLE2 chart servers provide a *generalized* chart interface for users to edit embedded charts. The end user must first in-place activate the chart and then invoke any number of chart dialogs to find and edit the appropriate chart properties. This interface may be acceptable in some instances. However, developers often would like the flexibility to provide a more intuitive and efficient interface. Oracle Graphics offers advantages over other OLE2 graphics servers by allowing the Forms developer to construct a highly *customized* interface for working with charts.

Change Axis Settings

Figure 6 shows a Forms application that allows the user to specify different axis settings for the embedded SmartChart. Separate text fields allow the user to specify axis minimum, maximum, and step values. This capability is often useful for end-users because at *design time* it is difficult for developers to anticipate the best axis settings for data that will be retrieved at *runtime*. This particular example demonstrates how a developer might create an interface for the end user to specify axis settings. However, the application developer can just as easily create an application that automatically adjusts the axis settings. By applying heuristics that determine the best axis settings, developers can ensure that any data set is presented properly at runtime. Forms developers have a great deal of flexibility to *control* embedded SmartCharts in the manner that best suits their application requirements.

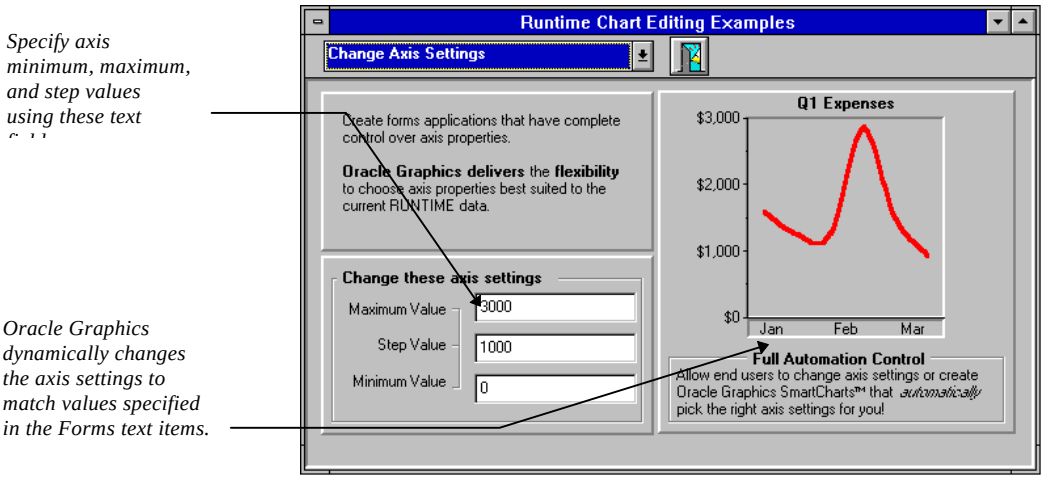

Figure 6.

Change Titles, Labels

The example shown in Figure 7 demonstrates one way a Forms application can allow end users to customize the titles and axis labels of an embedded SmartChart. As in the previous example, the application developer could easily have created an application that *automatically* changed these chart properties in response to the runtime context. For example, the chart title could be automatically derived from a runtime value retrieved from the database server.

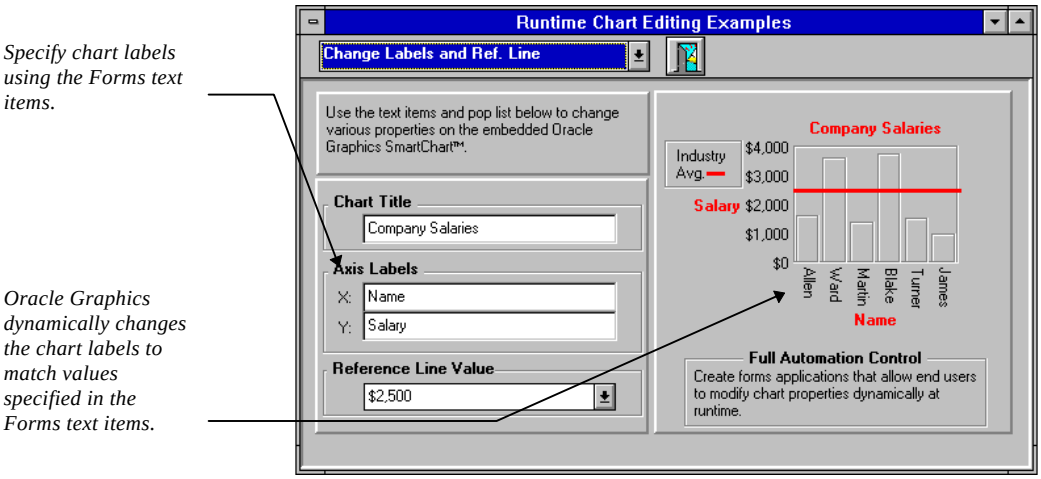

Figure 7.

Automatic Exception Reporting

Another benefit available to Forms developers is the ability to embed graphical objects that automatically highlight important information for the user. No other charting packages provide the amount of power and flexibility that Graphics delivers in this area. The following examples show different ways this feature can be put to effective use within Forms applications.

Highlighting Values Above/Below Thresholds

Users often need to scan data quickly and check if values fall above or below a specific threshold. Figure 8 shows an embedded SmartChart that colors salary bars red when an employee earns a salary above $4,000. Through Graphics' automation support, the Forms application can allow the user to specify a different salary threshold. When the user types a new value into the threshold text item, Forms instructs Graphics to adjust the highlighting criteria accordingly. This type of embedded SmartChart provides an effective format in which users can quickly identify important information.

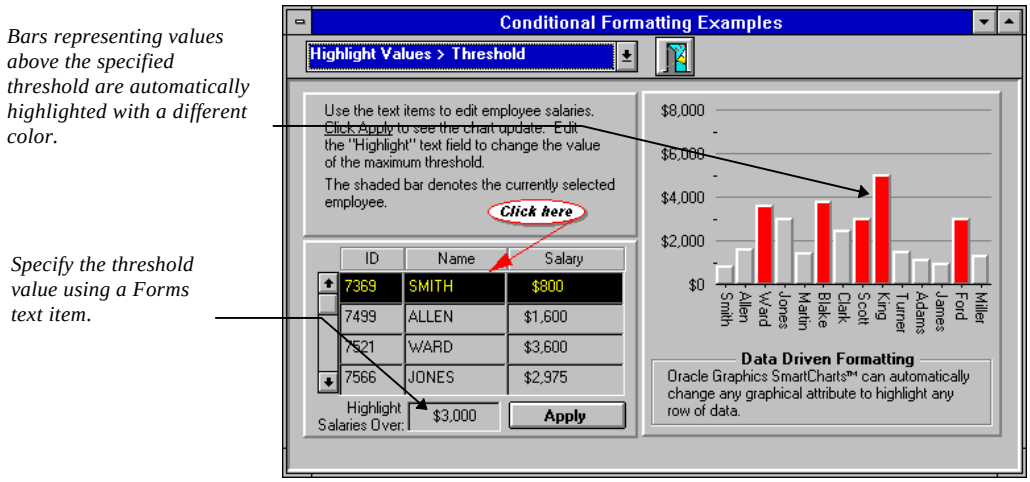

Figure 8.

Highlighting Max/Min Values

Forms developers may wish to apply other criteria for determining exceptions. Graphics allows developers to test for *any* condition when highlighting data. Figure 9 shows an example of highlighting the maximum value in the current data set. The user can use the Forms list item to browse salaries within different departments. When displaying employee salaries (for a given department), the embedded SmartChart automatically highlights the employee earning the highest salary by exploding their pie slice.

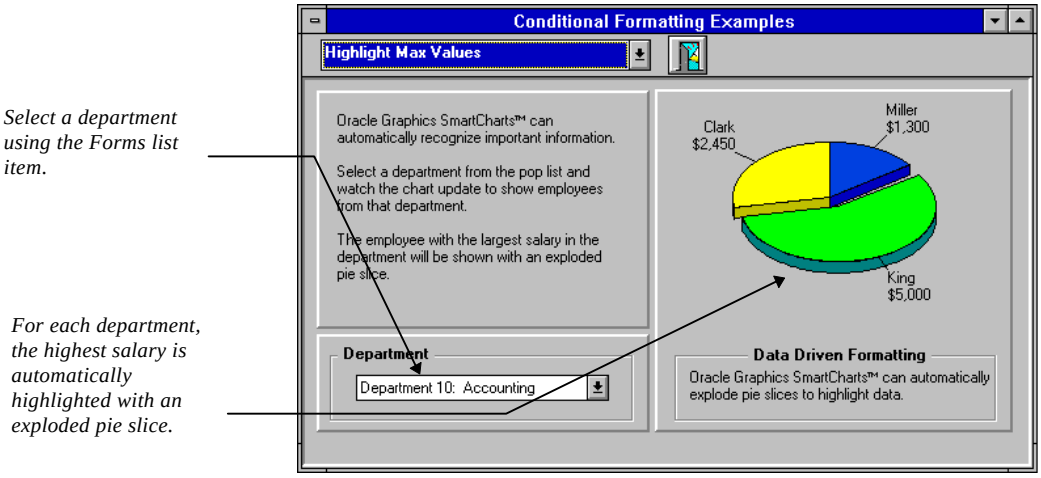

Figure 9.

Highlighting Increases

Another popular application of Graphics' conditional chart formatting is to highlight data values that have increased. For example, the embedded SmartChart in Figure 10 automatically highlights salaries that have increased by more than seven percent. Data values are highlighted by coloring the bars and by changing their bevel style. In general, Graphics allows developers to change *any* graphical property in order to highlight data.

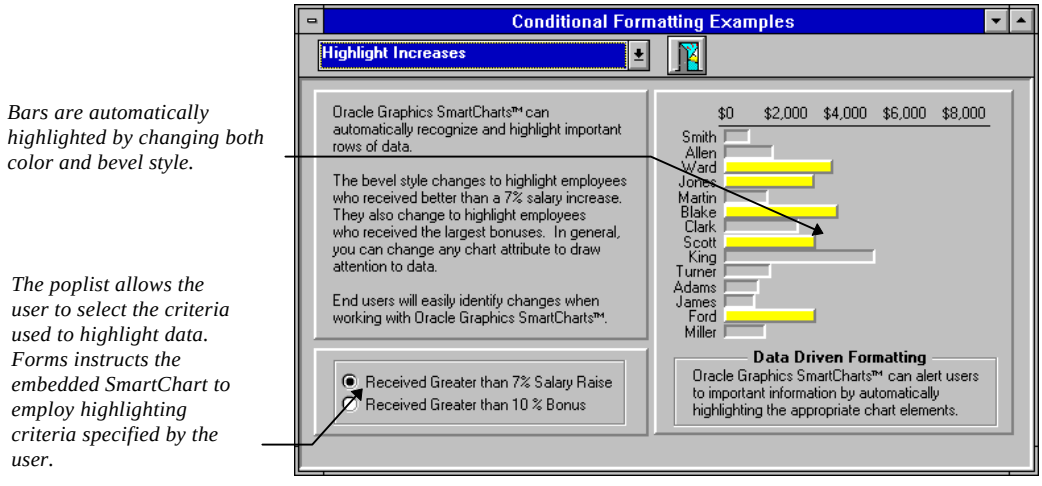

Figure 10.

Minimal Coding

The tight level of integration between Forms and Graphics significantly reduces the effort required to develop integrated applications. Many tasks, including data passing and chart updating, are handled automatically when embedding Graphics within Forms. Much *more code* is needed to integrate Forms with other graphics servers such as Microsoft Excel Chart.

The Forms application in the figure below shows information about managers and their direct reports.

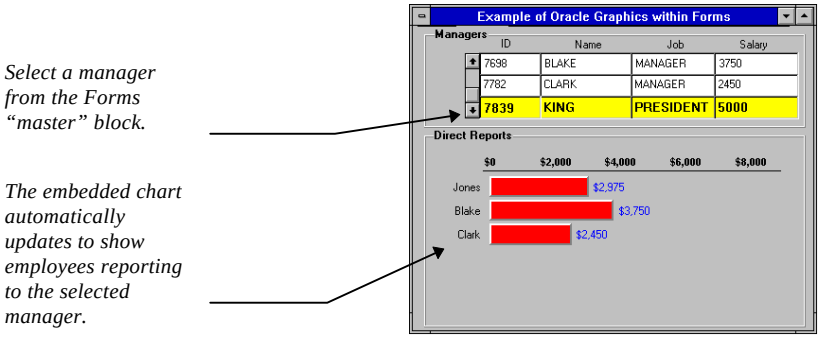

Figure 11.

The embedded Oracle Graphics SmartChart automatically updates to show salaries for employees reporting to the selected manager. This application was also created using Microsoft Excel Chart as an embedded OLE graphics server. The figure below provides a comparison of the PL/SQL code used to integrate Forms with Microsoft Excel Chart (via OLE) and the PL/SQL code used to integrate Forms with Graphics.

Code Required for *Excel Chart* Integration:

```
PROCEDURE init_Excel IS
   alist OLE2.List_Type;
   start_time number;
   end_time number;
BEGIN
   start_time := to_number(to_char(sysdate, 'ss'));
   Forms_OLE.Activate_Server('Excel_Chart');
   Global.Chart_Obj := Forms_OLE.Get_Interface_Pointer ('Excel_Chart');
   Global.XL := OLE2.Get_Obj_Property (Global.Chart_Obj, 'application');

   alist := ole2.create_arglist;
   ole2.add_arg(alist, 1);
   Global.WkBook := OLE2.Invoke_Obj (Global.XL, 'workbooks', alist);
   ole2.destroy_arglist(alist);
   alist := ole2.create_arglist;
   ole2.add_arg(alist, 1);
   Global.Data_Sheet := OLE2.Invoke_Obj (Global.WkBook, 'worksheets', alist);
   OLE2.Destroy_ArgList (alist);

   alist := ole2.create_arglist;
   ole2.add_arg(alist, 'chart_data');
   Global.Chart_Data_Range := OLE2.Invoke_Obj (Global.Data_Sheet, 'range', alist);
   OLE2.Destroy_ArgList (alist);
END;

PROCEDURE pass_block IS
   alist OLE2.List_Type;
   last_empno NUMBER;
   row_index number := 2;
   cell_obj OLE2.Obj_Type;
   wkbook OLE2.Obj_Type;
BEGIN
   OLE2.Invoke (Global.Chart_Data_Range, 'clear');
   Go_Block ('EMPS');
   Last_Record;
   last_empno := :EMPS.empno;
   First_Record;

   Loop
      alist := OLE2.Create_ArgList;
      OLE2.Add_Arg (alist, row_index);
      OLE2.Add_Arg (alist, 1);
      Cell_Obj := OLE2.Invoke_Obj (Global.Data_Sheet, 'Cells', alist);
      OLE2.Set_Property (Cell_Obj, 'Value', :EMPS.ENAME);
      OLE2.Destroy_ArgList (alist);
      OLE2.Release_Obj (Cell_Obj);

      alist := OLE2.Create_ArgList;
      OLE2.Add_Arg (alist, row_index);
      OLE2.Add_Arg (alist, 2);
      Cell_Obj := OLE2.Invoke_Obj (Global.Data_Sheet, 'cells', alist);
      OLE2.Set_Property (Cell_Obj, 'value', :EMPS.SAL);
      OLE2.Destroy_ArgList (alist);
      OLE2.Release_Obj (Cell_Obj);

      IF :EMPS.empno = last_empno THEN
         EXIT;
      END IF;
      Next_Record;
      row_index := row_index + 1;
   END LOOP;
   Go_Block ('MGR');
END;

PROCEDURE free_ole_objs IS
BEGIN
   OLE2.Release_Obj (Global.Chart_Data_Range);
   OLE2.Release_Obj (Global.Data_Sheet);
   OLE2.Release_Obj (Global.WkBook);
   OLE2.Release_Obj (Global.XL);
   Forms_OLE.Close_Server('Excel_Chart');
END;
```

Code Required for *Graphics* Integration:

```
PROCEDURE update_OG IS
  plist paramlist;
BEGIN
   plist := create_parameter_list ('parameter_list');
   Add_Parameter (plist, 'mgrid', TEXT_PARAMETER, to_char(:mgr.empno));
   OG.Interpret ('dirrep1.ogd', 'OG_Chart', 'update_chart;', plist=>plist);
   Destroy_Parameter_List (plist);
END;
```

Figure 12. Comparison of Code Used to Implement Example in Figure 11.

When integrating Forms with Excel Chart, PL/SQL must be written to pass data from the "master" block to the embedded chart ***one cell at time***. The resulting code cannot be easily reused in other Forms modules because it is very specific to the application data. Thus, even in the case of this simple example, developers must put significant effort into integrating Excel Chart into a Forms application.

Portability

Holding true to Oracle tradition, Forms and Graphics provide developers with a ***portable*** solution by delivering an internal integration mechanism that works on all major platforms.

Both Oracle Forms 4.5 and Oracle Graphics 2.5 provide OLE2 support for better integration with other Windows 3.1 applications. However, for many Forms developers, OLE2 does not offer an adequate integration solution because it is not portable. At the present time, Forms applications with embedded OLE2 servers cannot be used on other platforms besides Microsoft Windows. Thus, in many cases, portability concerns make SmartCharts the only viable solution for delivering graphical Forms applications.

Object Reuse

A significant feature of SmartCharts is their ability to *encapsulate* both behavior and data. Encapsulation hides much of the complexity and processing details, making embedded objects very easy to reuse once they have been defined.

As mentioned earlier, Forms developers have the capability of calling custom Graphics procedures from within Forms. This feature enables developers to create SmartCharts encapsulated with simple procedures for customizing appearance and behavior. For example, developers can exploit custom procedures supplied with the USA SmartChart to specify different *thresholds* for highlighting state regions. They can even specify different *colors* for highlighting regions. Thus, SmartChart modules are flexible enough to be reused by several *different* Forms applications. By treating SmartCharts as reusable objects, Forms developers can streamline their efforts significantly.

SmartCharts are *not only* reusable across Forms modules, they are also reusable across different OLE2 container applications. The same SmartCharts developed for integrated Forms applications can be easily reused within OLE2 containers such as Excel or Visual Basic. Companies that deal with multiple development environments can leverage Graphics' OLE2 support and eliminate redundant development efforts. For example, a *corporate* MIS group may use Forms to develop applications, while developers at the *departmental* level uses Visual Basic. Despite having different development tools, *both* teams can embed the *same* Oracle Graphics SmartCharts within their applications. The figure below shows the USA SmartChart (shown earlier embedded within Oracle Forms) embedded within Visual Basic.

Because *data* is encapsulated as part of the embedded object, the Visual Basic application did not need to establish a connection to the Oracle7 server in order to retrieve population data. Graphics can maintain its own Oracle7 connection while acting as an embedded graphics server. Thus, data encapsulation greatly enhances the reusability of embedded Graphics objects.

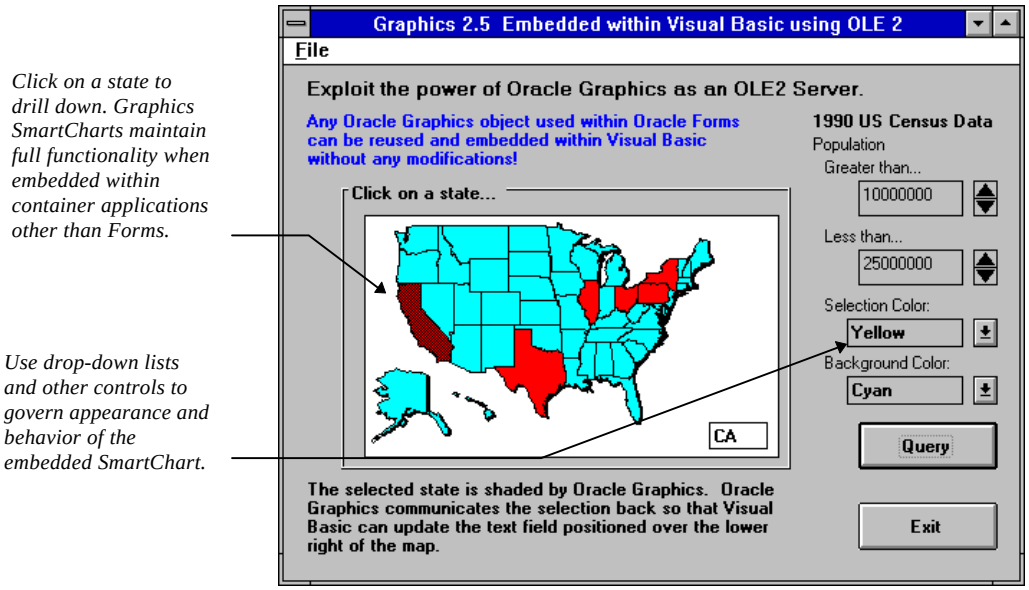

Figure 12.

Conclusion

Forms developers enjoy a wealth of new features with the new Developer/2000 product set. As the centerpiece of development for client-server applications, Oracle Forms offers an open environment for embedding various components. While there are many alternatives when choosing embedded objects, Oracle Graphics offers a clear choice for delivering powerful Forms applications. Graphics' strengths in the areas of graphical drill-down and automatic exception reporting are unsurpassed. Furthermore, no other product in today's market offers Forms developers the same level of integration and automation control as embedded Oracle Graphics SmartCharts. No other product offers Forms developers an integrated solution that is both open and portable. Oracle Graphics equips Forms developers with the ability to build a new class of applications that are more powerful than anything else previously developed using Oracle Forms.

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
USA

Worldwide Inquiries:
Phone: 415.506.8000
Fax: 415.506.7200

Author: Ivan Chong
Date: March, 1995